On Strong Normalization of the Calculus of Constructions with Type-Based Termination

Benjamin Grégoire, Jorge Luis Sacchini

On Strong Normalization of the Calculus of Constructions with Type-Based Termination

Benjamin Grégoire and Jorge Luis Sacchini

INRIA Sophia Antipolis - Méditerranée, France
{Benjamin.Gregoire,Jorge-Luis.Sacchini}@inria.fr

Abstract. Termination of recursive functions is an important property in proof assistants based on dependent type theories; it implies consistency and decidability of type checking. Type-based termination is a mechanism for ensuring termination that uses types annotated with size information to check that recursive calls are performed on smaller arguments. Our long-term goal is to extend the Calculus of Inductive Constructions with a type-based termination mechanism and prove its logical consistency. In this paper, we present an extension of the Calculus of Constructions (including universes and impredicativity) with sized natural numbers, and prove strong normalization and logical consistency. Moreover, the proof can be easily adapted to include other inductive types.

1 Introduction

Termination of recursive functions is an important property in proof assistants based on dependent type theory; it implies consistency of the logic, and decidability of type checking. In current implementations, it is common to use syntactic criteria (called guard predicates) to ensure termination. Guard predicates are applied to the body of recursive functions to check that recursive calls are performed on structurally smaller arguments. However, these criteria are often difficult to understand and implement.

An alternative approach is *type-based termination*. The basic idea is the use of sized types, i.e., types decorated with size information. Termination is ensured by typing constraints, restricting recursive calls to smaller arguments. Compared to guard predicates, type-based termination provides a simple yet expressive mechanism for ensuring termination.

In previous work by the first author [4], the Calculus of Inductive Constructions (CIC) was extended with a type-based termination mechanism. The obtained system, called CIC$\hat{}$, has many desirable metatheoretical properties, including complete size inference. However, the logical consistency of CIC$\hat{}$ is derived from a conjecture stating strong normalization of well-typed terms.

Our long-term goal is to define a type-based termination mechanism for CIC that is proved consistent. This paper is a step in that direction. We present a simplified version of CIC$\hat{}$, called CIC$\hat{}_{-}$ (Sect. 3).

Our main contribution is a proof of strong normalization (SN) and logical consistency for CIC$\hat{}$ restricted to one inductive type, namely natural numbers (Sect. 4). The interpretation of natural numbers is done in a generic way, and can be extended to other inductive types.

For lack of space, most of the proofs are omitted. The interested reader can find them in the long version of this paper [7].

2 A Primer on Type-based Termination

Before giving the details of CIC$\hat{}$, this section introduces briefly the main ideas of sized types and type-based termination. Consider the type of natural numbers, defined by

$$\mathsf{nat} : \mathsf{Type} := \mathsf{O} : \mathsf{nat} \mid \mathsf{S} : \mathsf{nat} \to \mathsf{nat} \ .$$

With sized types, this defines a family of types of the form nat^s, where s is a size (or stage) expression. Size information is used to type recursive functions, as shown in the following (simplified) typing rule for fixpoint construction:

$$\frac{\Gamma(f : \mathsf{nat}^\imath \to U) \vdash M : \mathsf{nat}^{\widehat{\imath}} \to U\,[\imath := \widehat{\imath}]}{\Gamma \vdash (\mathsf{fix}\ f : \mathsf{nat} \to U := M) : \mathsf{nat}^s \to U\,[\imath := s]}$$

where $\imath$ is a stage variable, and $\widehat{\cdot}$ is the successor function on stages. Intuitively, $\mathsf{nat}^\imath$ and $\mathsf{nat}^{\widehat{\imath}}$ denote the type of natural numbers whose size is smaller than $\imath$ and $\imath + 1$, respectively. This intuition is reflected in the subtyping rule, stating that $\mathsf{nat}^\imath \leq \mathsf{nat}^{\widehat{\imath}} \leq \mathsf{nat}^\infty$, where nat^∞ denotes the usual type of natural numbers.

In the typing rule above, the body of the fixpoint, M, is a function that takes an argument of type $\mathsf{nat}^{\widehat{\imath}}$. The recursive calls to f in M are only allowed on terms of type $\mathsf{nat}^\imath$, that is, smaller than the size of the argument. This ensures that recursion terminates. Note that the variable $\imath$ can appear (positively) in U, which allows to write size-preserving functions. A typical example is the subtraction of natural numbers, which has type $\mathsf{nat}^s \to \mathsf{nat}^\infty \to \mathsf{nat}^s$, for any s. We can then write the division of natural numbers as[1]:

$$\begin{array}{l}\mathsf{fix\ div} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} := \\ \quad \lambda\ m\ n : \mathsf{nat}.\ \mathsf{case}\ m_{\mathsf{nat}(\widehat{\imath})}\ \mathsf{of} \\ \qquad\qquad \mid \mathsf{O} \Rightarrow \mathsf{O}_{\mathsf{nat}(\widehat{\imath})} \\ \qquad\qquad \mid \mathsf{S}\ m'_{\mathsf{nat}(\imath)} \Rightarrow S\ (\mathsf{div}\ (\mathsf{minus}\ m'\ n)_{\mathsf{nat}(\imath)}\ n)_{\mathsf{nat}(\imath)}\end{array}$$

The type annotations are given for clarification only, and are not written in the actual syntax. Since minus has type $\mathsf{nat}^s \to \mathsf{nat}^\infty \to \mathsf{nat}^s$, the recursive call in div is well-typed. The function div also has type $\mathsf{nat}^s \to \mathsf{nat}^\infty \to \mathsf{nat}^s$.

Note in the case expression that m has type $\mathsf{nat}^{\widehat{\imath}}$, while the recursive argument m' has a smaller type $\mathsf{nat}^\imath$. This mechanism ensures that we can make recursive calls on the recursive arguments of an inductive type. However, it is

[1] div m n computes $\left\lceil \frac{m}{n+1} \right\rceil$

more powerful than guard predicates, thanks to the possibility of typing size-preserving functions. For example, extending the system with sized lists, typical functions can be given more precise types:

$$\mathsf{map} : \Pi(A\ B : \mathsf{Type}).(A \to B) \to \mathsf{list}^{\imath}\ A \to \mathsf{list}^{\imath}\ B$$
$$\mathsf{filter} : \Pi(A : \mathsf{Type}).(A \to \mathsf{bool}) \to \mathsf{list}^{\imath}\ A \to \mathsf{list}^{\imath}\ A \times \mathsf{list}^{\imath}\ A$$

These functions allow to type programs that are not structurally recursive such as quicksort:

$$\begin{array}{l}\mathsf{fix}\ \mathsf{qsort} : \mathsf{list}\ A \to \mathsf{list}\ A := \\ \lambda l : \mathsf{list}\ A.\ \mathsf{case}\ l_{\mathsf{list}(\widehat{\imath})}\ \mathsf{of} \\ \qquad\qquad \mid \mathsf{nil} \Rightarrow \mathsf{nil} \\ \qquad\qquad \mid \mathsf{cons}\ h\ t_{\mathsf{list}(\imath)} \Rightarrow \mathsf{let}\ (s,g) = \mathsf{filter}\ (<h)\ t_{\mathsf{list}(\imath)}\ \mathsf{in} \\ \qquad\qquad\qquad\qquad\qquad \mathsf{append}\ (\mathsf{qsort}\ s_{\mathsf{list}(\imath)})\ (\mathsf{cons}\ h\ (\mathsf{qsort}\ g_{\mathsf{list}(\imath)}))\end{array}$$

Note that the precise typing of filter allows the recursive calls in the above definition. However, in this case, qsort has type $\mathsf{list}^{\infty}A \to \mathsf{list}^{\infty}A$. For further examples and references, we refer the reader to [4].

3 System CIC$^{\widehat{\ }}_{-}$

CIC$^{\widehat{\ }}_{-}$ is an extension of CIC with a type-based termination mechanism. In this section we introduce the syntax and typing rules.

In order to treat impredicativity in the proof of SN, terms carry more type annotations in the case of abstraction and application [2,11]. However, the system we intend to use has a more traditional presentation. In the traditional presentation, abstractions have the form $\lambda x : T^{\circ}.M$ and applications have the form $M\,N$. We give here the annotated presentation we use in the proof of SN. The reduction rules and typing rules are adapted to the traditional presentation in the obvious way. Note that as a consequence of SN, we can prove the equivalence between both presentations [7].

Syntax. We consider a predicative hierarchy of sorts Type_i, for $i \in \mathbb{N}$, and an impredicative sort Prop. The set of sorts is denoted $\mathcal{U}$. We assume a denumerable set $\mathcal{X}$ of term variables. We use u to denote sorts, and f, x, y to denote term variables. Inductive types are annotated with stage expressions.

Definition 1 (Stages). *The syntax of stages is given by the following grammar, where $\mathcal{X}_{\mathcal{S}}$ denotes a denumerable set of stage variables.*

$$\mathcal{S} ::= \mathcal{X}_{\mathcal{S}} \mid \widehat{\mathcal{S}} \mid \infty$$

We use $\imath$, $\jmath$ to denote stage variables, and s, r to denote stages. The base of a stage expression is defined by $\lfloor \imath \rfloor = \imath$ and $\lfloor \widehat{s} \rfloor = \lfloor s \rfloor$ (the base of a stage containing ∞ is not defined).

The syntax features three classes of terms, whose difference lies in the type of annotations that inductive types carry. (This helps to ensure subject reduction and efficient type inference [4].) Bare terms carry no annotation. Position terms carry either no annotation, or a $\star$, which is used to indicate recursive positions in fixpoint functions. Finally, sized terms carry a stage expression.

Definition 2 (Terms). *The generic set of terms over the set a is defined by the grammar:*

$$\begin{aligned}\mathcal{T}[a] ::= {} & \mathcal{X} \mid \mathcal{U} \mid \lambda_{\mathcal{X}:\mathcal{T}^\circ.\mathcal{T}^\circ}\mathcal{T}[a] \mid \mathsf{app}_{\mathcal{X}:\mathcal{T}^\circ.\mathcal{T}^\circ}(\mathcal{T}[a],\mathcal{T}[a]) \mid \Pi\mathcal{X}:\mathcal{T}[a].\mathcal{T}[a] \\ & \mid \mathsf{nat}^a \mid \mathsf{O} \mid \mathsf{S}(\mathcal{T}[a]) \\ & \mathsf{case}_{\mathcal{T}^\circ}\ \mathcal{X} := \mathcal{T}[a]\ \mathsf{of}\ \mathcal{T}[a],\mathcal{T}[a] \mid \mathsf{fix}\ \mathcal{X}(\mathcal{X}:\mathsf{nat}^\star):\mathcal{T}^\star := \mathcal{T}[a]\end{aligned}$$

The set of bare terms, position terms and sized terms are defined by $\mathcal{T}^\circ ::= \mathcal{T}[\epsilon]$, $\mathcal{T}^\star ::= \mathcal{T}[\{\epsilon,\star\}]$, and $\mathcal{T} ::= \mathcal{T}[\mathcal{S}]$, respectively. We use M, N, P, T, U, C to denote terms. Bare terms are usually denoted with a superscript $\circ$ and position terms with a superscript $\star$, as in M° and $M^\star$.

To deal with the different classes of terms, we use two erasure functions: the function $|.| : \mathcal{T}^\star \cup \mathcal{T} \to \mathcal{T}^\circ$ removes all annotations from a term; the function $|.|^\imath : \mathcal{T} \to \mathcal{T}^\star$ replaces all stage annotations s with $\star$ if $\lfloor s \rfloor = \imath$, or by ϵ otherwise. Given a term M, we write M^∞ to denote the term M where all size annotations are replaced with ∞, and $\mathrm{SV}(M)$ to denote the set of *stage variables* appearing in M.

Definition 3. *Reduction $\to$ is defined as the compatible closure of β-reduction, ι-reduction and μ-reduction:*

$$\begin{aligned}\mathsf{app}_{x:T^\circ.U^\circ}(\lambda_{x:T^\circ.U^\circ}M, N) &\to_\beta M\,[x := N] \\ \mathsf{case}_{T^\circ}\ x := \mathsf{O}\ \mathsf{of}\ N_0, N_1 &\to_\iota N_0 \\ \mathsf{case}_{T^\circ}\ x := \mathsf{S}(M)\ \mathsf{of}\ N_0, N_1 &\to_\iota \mathsf{app}_{y:\mathsf{nat}.T^\circ[x:=S(y)]}(N_1, M) \\ \mathsf{app}_{x:\mathsf{nat}.|U^\star|}(F, C) &\to_\mu \mathsf{app}_{x:\mathsf{nat}.|U^\star|}(M\,[f := F], C)\end{aligned}$$

where $F \equiv \mathsf{fix}\ f(x:\mathsf{nat}^\star):U^\star := M$ and C is a term in constructor form (i.e., O or $\mathsf{S}(M)$ for some term M). We write $\leftarrow$, $\to^*$, $\approx$ and $\downarrow$ for the inverse relation, the reflexive transitive closure, the equivalence closure and the associated joinability relation of $\to$, respectively. ($M \downarrow N$ if $M \to^* P$ and $N \to^* P$ for some P.)

The reduction relation defined above, usually called *tight reduction*, is *not* confluent (on pseudoterms). However, is it confluent for well-typed terms (this is a consequence of SN). (Note that the reduction in the traditional presentation is confluent.)

Subtyping. We consider a subtyping relation derived from a partial order on stages.

Definition 4 (Substage). *The* substage relation, *$\sqsubseteq \subseteq \mathcal{S} \times \mathcal{S}$, is defined as the reflexive transitive closure of the relation containing $s \sqsubseteq \widehat{s}$ and $s \sqsubseteq \infty$, for all $s \in \mathcal{S}$.*

Definition 5 (Subtyping). *The* subtyping relation, $\leq \subseteq \mathcal{T} \times \mathcal{T}$, *is defined by following rules:*

$$\frac{T \downarrow U}{T \leq U} \qquad \frac{T \to^* \mathsf{nat}^s \qquad U \to^* \mathsf{nat}^r \qquad s \sqsubseteq r}{T \leq U}$$

$$\frac{T \to^* \Pi x : T_1.T_2 \qquad U \to^* \Pi x : U_1.U_2 \qquad U_1 \leq T_1 \qquad T_2 \leq U_2}{T \leq U}$$

We define a notion of positivity with respect to a stage variable. It is used in the typing rules to restrict the types valid for recursion.

Definition 6. *We say that* $\imath$ is positive in T, *written* $\imath$ pos T, *if for every pair of stages* s, r, *such that* $s \sqsubseteq r$, $T[\imath := s] \leq T[\imath := r]$.

Remark: In the traditional presentation, the subtyping relation is defined in a more standard way, as the reflexive transitive closure of the following rules:

$$\frac{T \approx U}{T \leq U} \qquad \frac{U_1 \leq T_1 \qquad T_2 \leq U_2}{\Pi x : T_1.T_2 \leq \Pi x : U_1.U_2} \qquad \frac{s \sqsubseteq r}{\mathsf{nat}^s \leq \mathsf{nat}^r}$$

We cannot use this definition in the annotated presentation, since reduction is not confluent.

Typing. In the typing rules, we restrict the use of size variables appearing in types. Intuitively, we only allow types that reduce to a term of the form

$$\Pi x_1 : T_1.\Pi x_2 : T_2. \ldots \Pi x_n : T_n.T_{n+1}, \tag{*}$$

where each T_i is of the form (*), or is of the form nat^s, or satisfies $\mathrm{SV}(T_i) = \emptyset$. We call these types "simple". Formally, we define a predicate simple with the following clauses:

$$\frac{\mathrm{SV}(T) = \emptyset}{\mathsf{simple}(T)} \qquad \frac{}{\mathsf{simple}(\mathsf{nat}^s)} \qquad \frac{\mathsf{simple}(T_1) \qquad \mathsf{simple}(T_2)}{\mathsf{simple}(\Pi x : T_1.T_2)}$$

Contexts and judgments. A context is a sequence of the form $(x_1 : T_1)(x_2 : T_2) \ldots (x_n : T_n)$, where $x_1, x_2, \ldots, x_n$ are distinct variables and $T_1, T_2, \ldots, T_n$ are sized terms. We use Γ, Δ, Θ to denote contexts and [] to denote the empty context.

We define two typing judgments: $\mathsf{WF}(\Gamma)$ means that context Γ is well formed; $\Gamma \vdash M : T$ means that the term M has type T in Γ. The typing rules are given in Fig. 1. The side conditions in some of the rules ensure that we restrict to simple types. If we remove these side conditions, the resulting system is that of CIC$\hat{}$. In rule (fix) the condition $\imath \notin \mathrm{SV}(\Gamma, M)$ is therefore redundant, but we keep it to emphasize the difference with CIC$\hat{}$.

Most of the rules are similar to that of CIC$\hat{}$, with exception of the added type annotations. Note in rules (zero) and (succ) that constructors have a successor

stage as type. In rule (case), as we mentioned in Sect. 2, the argument has type nat with a successor stage, allowing the recursive arguments to have smaller size. Note that, because of subtyping, any term of type nat can be given a successor size. In rule (fix) we introduce a fresh size variable for recursion. Not every type is valid for recursion, since it might lead to inconsistencies [1]. In our case, we require the size variable used for recursion to appear positively in the return type. The body, M, has a product type (function) with domain $\mathsf{nat}^{\widehat{\imath}}$, while recursive calls to f can only be performed on terms of type $\mathsf{nat}^{\imath}$.

Simple metatheory. Usual metatheoretic results such as weakening, substitution and subject reduction can be proved for CIC$^{\widehat{}}_{\sqsubseteq}$ in the same way as for CIC$^{\widehat{}}$. These properties are stated in Fig. 2.

4 Strong Normalization

In this section we prove the main results of the paper: strong normalization of CIC$^{\widehat{}}_{\sqsubseteq}$, and logical consistency (Theorem 1). The proof is based on Λ-sets as introduced by Altenkirch in his PhD thesis [2], and later used by Melliès and Werner [11] to prove strong normalization for Pure Type Systems.

A Λ-set X is a pair $(X_\circ, \models)$, where $X_\circ$ is a set, and $\models \subseteq \mathrm{SN} \times X_\circ$ is a realizability relation[2] (SN denotes the set of strongly normalizing terms). Intuitively, we define a set-theoretical interpretation (products are interpreted by function spaces, abstractions by functions and applications by function application), denoted $[\cdot]$, corresponding to the set part of a Λ-set.

We prove that the interpretation is sound: if $\Gamma \vdash M : T$, then $[M] \in [T]$ (Lemma 4). We can then prove that every term realizes its interpretation (Lemma 5), i.e. $M \models [M]$. Strong normalization (Corollary 1) follows from the fact that every realizers is strongly normalizing by definition.

In the case of CIC$^{\widehat{}}_{\sqsubseteq}$, the interpretation given above does not take size information into account. We therefore define a second (relational) interpretation to show that terms respect the size information given in the type (Sect. 4.3).

4.1 Preliminary Definitions

In this section we give the concepts necessary to define the interpretation of terms. Namely, saturated sets, Λ-sets, and inaccessible cardinals.

Saturated sets. We define saturated sets in terms of elimination contexts:

$$\begin{aligned} E[] ::= {} & [] \mid \mathsf{app}_{\mathcal{X}:\mathcal{T}^\circ.\mathcal{T}^\circ}(E[], \mathcal{T}) \mid \mathsf{case}_{\mathcal{T}^\circ}\ \mathcal{X} := E[]\ \mathsf{of}\ \mathcal{T}, \mathcal{T} \\ & \mid \mathsf{app}_{\mathcal{X}:\mathcal{T}^\circ.\mathcal{T}^\circ}(\mathsf{fix}\ \mathcal{X}(\mathcal{X} : \mathsf{nat}^\star) : \mathcal{T}^\star := \mathcal{T}, E[]) \end{aligned}$$

A term is *atomic* if it is of the form $E[x]$. We denote the set of atomic terms with AT, and the set of strongly normalizing terms with SN. Weak-head reduction is defined as $E[M] \to_{\mathrm{wh}} E[N]$ iff $M \to_{\beta\iota\mu} N$.

[2] Actually, for technical reasons, our definition is slightly different.

$$\text{(empty)}\quad \frac{}{\mathsf{WF}([])} \qquad \text{(cons)}\quad \frac{\mathsf{WF}(\Gamma) \qquad \Gamma \vdash T : u}{\mathsf{WF}(\Gamma(x:T))}\ \mathsf{simple}(T)$$

$$\text{(var)}\quad \frac{\mathsf{WF}(\Gamma) \qquad \Gamma(x) = T}{\Gamma \vdash x : T}$$

$$\text{(type)}\quad \frac{\mathsf{WF}(\Gamma)}{\Gamma \vdash \mathsf{Type}_i : \mathsf{Type}_{i+1}} \qquad \text{(prop)}\quad \frac{\mathsf{WF}(\Gamma)}{\Gamma \vdash \mathsf{Prop} : \mathsf{Type}_0}$$

$$(\Pi\text{-type})\quad \frac{\Gamma \vdash T : u \qquad \Gamma(x:T) \vdash U : \mathsf{Type}_j}{\Gamma \vdash \Pi x : T.U : \max(u, \mathsf{Type}_j)}$$

$$(\Pi\text{-prop})\quad \frac{\Gamma \vdash T : u \qquad \Gamma(x:T) \vdash U : \mathsf{Prop}}{\Gamma \vdash \Pi x : T.U : \mathsf{Prop}}$$

$$\text{(abs)}\quad \frac{\Gamma(x:T) \vdash M : U}{\Gamma \vdash \lambda_{x:|T|.|U|} M : \Pi x : T.U}\ \mathrm{SV}(M) = \emptyset$$

$$\text{(app)}\quad \frac{\Gamma \vdash M : \Pi x : T.U \qquad \Gamma \vdash N : T}{\Gamma \vdash \mathsf{app}_{x:|T|.|U|}(M, N) : U\,[x := N]}\ \mathrm{SV}(N) = \emptyset$$

$$\text{(nat)}\quad \frac{\mathsf{WF}(\Gamma)}{\Gamma \vdash \mathsf{nat}^s : \mathsf{Type}_0} \qquad \text{(zero)}\quad \frac{\mathsf{WF}(\Gamma)}{\Gamma \vdash \mathsf{O} : \mathsf{nat}^{\widehat{s}}} \qquad \text{(succ)}\quad \frac{\Gamma \vdash M : \mathsf{nat}^s}{\Gamma \vdash \mathsf{S}(M) : \mathsf{nat}^{\widehat{s}}}$$

$$\text{(case)}\quad \frac{\begin{array}{c}\Gamma \vdash M : \mathsf{nat}^{\widehat{s}} \qquad \Gamma(x : \mathsf{nat}^{\widehat{s}}) \vdash P : u \\ \Gamma \vdash N_0 : P\,[x := \mathsf{O}] \qquad \Gamma \vdash N_1 : \Pi y : \mathsf{nat}^s.P\,[x := \mathsf{S}(y)]\end{array}}{\Gamma \vdash \mathsf{case}_{|P|}\ x := M\ \mathsf{of}\ N_0, N_1 : P\,[x := M]}\ \mathrm{SV}(M) = \emptyset$$

$$\text{(fix)}\quad \frac{\begin{array}{c}T \equiv \Pi(x : \mathsf{nat}^{\imath}).U \qquad \imath\ \mathsf{pos}\ U \qquad \imath \notin \mathrm{SV}(\Gamma, M) \\ \Gamma \vdash T : u \qquad \Gamma(f : T) \vdash M : T\,[\imath := \widehat{\imath}]\end{array}}{\Gamma \vdash \mathsf{fix}\ f(x : \mathsf{nat}^{\star}) : |U|^{\imath} := M : T\,[\imath := s]}\ \mathrm{SV}(M) = \emptyset$$

$$\text{(conv)}\quad \frac{\Gamma \vdash M : T \qquad \Gamma \vdash U : u \qquad T \leq U}{\Gamma \vdash M : U}\ \mathsf{simple}(U)$$

Fig. 1. Typing rules of CIC$\widehat{\ }_{-}$

$$\text{Weakening: } \Gamma \vdash M : T \wedge \mathsf{WF}(\Gamma\Delta) \Rightarrow \Gamma\Delta \vdash M : T$$

$$\text{Substitution: } \left.\begin{array}{r}\Gamma(x:T)\Delta \vdash M : U \\ \Gamma \vdash N : T \\ \mathrm{SV}(N) = \emptyset\end{array}\right\} \Rightarrow \Gamma\Delta\,[x := N] \vdash M\,[x := N] : U\,[x := N]$$

$$\text{Stage Substitution: } \Gamma \vdash M : T \Rightarrow \Gamma\,[\imath := s] \vdash M\,[\imath := s] : T\,[\imath := s]$$

$$\text{Subject Reduction: } \Gamma \vdash M : T \wedge M \to M' \Rightarrow \Gamma \vdash M' : T$$

$$\text{Type validity: } \Gamma \vdash M : T \Rightarrow \Gamma \vdash T : u \wedge \mathsf{simple}(\mathrm{T})$$

Fig. 2. Simple metatheory of CIC$\widehat{\ }_{-}$

Definition 7 (Saturated set). *A set of terms $X \subseteq \mathrm{SN}$ is saturated iff it satisfies the following conditions:*

(S1) $\mathrm{AT} \cap \mathrm{SN} \subseteq X$;
(S2) if $M \in \mathrm{SN}$ *and* $M \to_{\mathrm{wh}} M'$ *and* $M' \in X$, *then* $M \in X$.

Λ-sets. As mentioned above, we use Λ-sets [2,11] in the proof. However, our definition is slightly different, as explained below.

Definition 8 (Λ-set). *A Λ-set is a triple $X = (X_\circ, \models_X, \bot_X)$ where $X_\circ$ is a non-empty set, witnessed by $\bot_X \in X_\circ$, and $\models_X \subseteq \mathcal{T} \times X_\circ$.*

$X_\circ$ is the *carrier-set* and the elements of $X_\circ$ are called the *carriers* of X. The terms M such that $M \models_X \alpha$ for some $\alpha \in X_\circ$ are called the *realizers* of α. The element $\bot_X$ is called the *atomic element* of X. We write $\alpha \in X$ for $\alpha \in X_\circ$. A Λ-set X is included in a Λ-set Y, written $X \subseteq Y$, if $X_\circ \subseteq Y_\circ$, $\models_X \subseteq \models_Y$, and $\bot_X = \bot_Y$.

Definition 9 (Saturated Λ-set). *A Λ-set X is said to be* saturated *if*

1. *every realizer is strongly normalizable;*
2. *the atomic element $\bot_X$ is realized by any atomic strongly normalizable term;*
3. *for every $\alpha \in X_\circ$, if $N \models_X \alpha$, and $M \to_{\mathrm{wh}} N$ with $M \in \mathrm{SN}$, then $M \models_X \alpha$ (i.e., the set of realizers is closed under weak-head expansion).*

The difference between the definition in [2,11] and ours is that the atomic element of a Λ-set is explicit in the definition. The use of the atomic element will be evident in the definition of the interpretation of terms. However, the difference in the definition is not essential in our proof.

We define some operations on Λ-sets.

Definition 10 (Product). *Let X be a Λ-set and $\{Y_\alpha\}_{\alpha \in X_\circ}$ a $X_\circ$-indexed family of Λ-sets. We define the Λ-set $\Pi(X, Y)$ by:*

- $\Pi(X,Y)_\circ = \{f \in X_\circ \to \bigcup_{\alpha \in X_\circ} (Y_\alpha)_\circ : \forall \alpha \in X_\circ . f(\alpha) \in (Y_\alpha)_\circ\}$;
- $M \models_{\Pi(X,Y)} f \iff \forall \alpha \in X_\circ .\ T^\circ, U^\circ \in \mathrm{SN}.$
  $N \models_X \alpha \Rightarrow \mathsf{app}_{x:T^\circ.U^\circ}(M, N) \models_{Y_\alpha} f(\alpha)$;
- $\bot_{\Pi(X,Y)} = \alpha \in X_\circ \mapsto \bot_{Y_\alpha}$.

Lemma 1. *If X and every $\{Y_\alpha\}_{\alpha \in X_\circ}$ are saturated Λ-sets, so is $\Pi(X, Y)$.*

Definition 11 (Cartesian product). *Let X, Y be Λ-sets. We define the Λ-set $X \times Y$ by: $(X \times Y)_\circ = X_\circ \times Y_\circ$; $M \models_{X \times Y} (\alpha, \beta) \iff M \models_X \alpha \wedge M \models_Y \beta$; and $\bot_{X \times Y} = (\bot_X, \bot_Y)$.*

We write X^2 for $X \times X$.

Lemma 2. *If X and Y are saturated Λ-sets, so is $X \times Y$.*

Definition 12 (Λ-iso). *Let X and Y be Λ-sets. A Λ-iso f from X to Y is a one-to-one function $f : X_\circ \to Y_\circ$ such that $M \models_X \alpha \iff M \models_Y f(\alpha)$, and $f(\bot_X) = \bot_Y$.*

Inaccessible cardinals. We assume an increasing sequence of inaccessible cardinals $\{\lambda_i\}_{i\in\mathbb{N}}$. Let V_α be the cumulative hierarchy of sets. We define $\mathcal{U}_i$ to be the set of saturated Λ-set whose carrier-set are in V_{λ_i}. The set $\mathcal{U}_i$ can be viewed as a Λ-set $(\mathcal{U}_i, \mathrm{SN}\times\mathcal{U}_i, \{\emptyset\})$[3]. Following [10], we interpret the predicative sorts using large universes.

4.2 The Interpretation

Stages. Stages are interpreted by ordinals. We use $\mathfrak{a}$, $\mathfrak{b}$, ... to denote ordinals. Since we have only natural numbers as a sized type, we can safely interpret stages with the smallest infinite ordinal, ω. If we include higher-order sized types (such as well-founded trees), we need to interpret stages using higher ordinals.

Definition 13 (Stage interpretation). *A stage assignment π is a function from $\mathcal{X}_\mathcal{S}$ to ω. Given a stage assignment π, the interpretation of a stage s under π, written $(\!|s|\!)_\pi$, is defined by:*

$$(\!|\imath|\!)_\pi = \pi(\imath), \qquad (\!|\infty|\!)_\pi = \omega, \qquad (\!|\widehat{s}|\!)_\pi = (\!|s|\!)_\pi \mathbin{\hat{+}} 1$$

where $\mathfrak{a} \mathbin{\hat{+}} 1 = \mathfrak{a} + 1$ if $\mathfrak{a} < \omega$, and $\omega \mathbin{\hat{+}} 1 = \omega$.

We use ∞ to denote the stage assignment such that $\infty(\imath) = \omega$ for all $\imath$.

Inductive types. Inductive types are interpreted as the least fixed point of a monotone operator. For our case, we define a function $\mathcal{F}_\mathcal{N}$, such that if X is a Λ-set, $\mathcal{F}_\mathcal{N}(X)$ is also a Λ-set defined by:

- $\mathcal{F}_\mathcal{N}(X)_\circ = \{\emptyset\} \cup \{(0,\emptyset)\} \cup \{(1,\alpha) : \alpha \in X_\circ\}$;
- $M \models_{\mathcal{F}_\mathcal{N}(X)} \alpha$, with $M \in \mathrm{SN}$, iff one the following conditions holds:
  - $\alpha = \emptyset$ and $M \to_{\mathrm{wh}}{}^* N \in \mathrm{AT}$;
  - $\alpha = (0,\emptyset)$ and $M \to_{\mathrm{wh}}{}^* \mathsf{O}$; or
  - $\alpha = (1,\alpha')$ and $M \to_{\mathrm{wh}}{}^* \mathsf{S}(M')$ with $M' \models_X \alpha'$.
- $\bot_{\mathcal{F}_\mathcal{N}(X)} = \emptyset$,

It is clear that if X is a saturated Λ-set, then $\mathcal{F}_\mathcal{N}(X)$ is also a saturated Λ-set. Note that $\mathcal{F}_\mathcal{N}$ is monotone, in the sense that if $X \subseteq Y$, then $\mathcal{F}_\mathcal{N}(X) \subseteq \mathcal{F}_\mathcal{N}(Y)$. We write $\mathcal{F}^k_\mathcal{N}$ to mean function $\mathcal{F}_\mathcal{N}$ iterated k times.

Consider the Λ-set $\bot = (\{\emptyset\}, \mathrm{SN}\cap\mathrm{AT}\times\{\emptyset\}, \emptyset)$. A fixpoint of $\mathcal{F}_\mathcal{N}(X)$ is reached by iterating ω times, starting from $\bot$. Let $\mathcal{N} = \mathcal{F}^\omega_\mathcal{N}(\bot)$.

Impredicativity. Following [11], we interpret the impredicative universe as the set of degenerated Λ-sets.

Definition 14. *A Λ-set X is* degenerated*, if the carrier-set $X_\circ$ is a singleton $\{A\}$, where A is a saturated set, $M \models_X A$ iff $M \in A$, and $\bot_X = A$.*

We write $\overline{A}$ for the degenerated Λ-set corresponding to a saturated set A.

[3] We choose $\{\emptyset\}$ as atomic element, but any element of $\mathcal{U}_i$ will do.

A proposition, i.e. a term T of type Prop, is represented by a degenerated Λ-set, whose only carrier represents a (possible) canonical proof.

Given a Λ-set X and a function Y such that for each $\alpha \in X_\circ$, Y_α is a degenerated Λ-set (with carrier y_α), the carrier-set of $\Pi(X,Y)$ is a singleton (the only element being $\alpha \in X_\circ \mapsto y_\alpha$). The canonical representation of $\Pi(X,Y)$ is given by the degenerated Λ-set corresponding to the saturated set

$$\downarrow(X,Y) = \{M \in \mathrm{SN} : N \Vdash_X \alpha \Rightarrow \mathsf{app}_{x:T^\circ.U^\circ}(M,N) \Vdash_{Y_\alpha} y_\alpha\} \ .$$

Note that there is a Λ-iso $p(X,Y) : \Pi(X,Y) \to \overline{\downarrow(X,Y)}$.

In the interpretation, we need to convert between the interpretation of a proof term as an element of $\Pi(X,Y)$ (if it needs to be applied), or as the canonical proof $\downarrow(X,Y)$. For this, we use the isomorphism $p(X,Y)$.

We define the functions $\Pi_{\Gamma\vdash T}$, $\downarrow_{\Gamma\vdash T}$, $\uparrow_{\Gamma\vdash T}$ that give the interpretation of products, abstractions, and applications (respectively), depending if the type T is a proposition or not. In the case of proposition, these functions convert between $\Pi(X,Y)$ and the canonical representation $\downarrow(X,Y)$. Otherwise, there is no conversion needed. Their definition is given by:

- if $\Gamma^\infty \vdash T^\infty : \mathsf{Prop}$, then $\Pi_{\Gamma\vdash T}(X,Y) = \overline{\downarrow(X,Y)}$, $\downarrow_{\Gamma\vdash T}(X,Y) = p(X,Y)$, and $\uparrow_{\Gamma\vdash T}(X,Y) = p^{-1}(X,Y)$;
- otherwise, $\Pi_{\Gamma\vdash T}(X,Y) = \Pi(X,Y)$, $\downarrow_{\Gamma\vdash T}(X,Y) = \mathrm{id}_{\Pi(X,Y)}$, and $\uparrow_{\Gamma\vdash T}(X,Y) = \mathrm{id}_{\Pi(X,Y)}$.

Terms and contexts. The interpretation of an erased context Γ is denoted $[\Gamma]$ (an erased context is a context formed by erased terms). Assume a stage assignment π. Given an erased context Γ, a term M and a sequence of values γ, the interpretation of M under Γ is denoted $[\Gamma \vdash M]^\pi_\gamma$.

We define the interpretation by induction on the structure of terms and contexts. In the case of fixpoint construction we use Hilbert's choice operator.

Definition 15 (Interpretation of terms and contexts).

$$\begin{aligned}
[[]] &= \{\emptyset\} \\
[\Gamma(x:T)] &= \{(\gamma,\alpha) : \gamma \in [\Gamma] \wedge \alpha \in [\Gamma \vdash T^\infty]^\infty_\gamma\} \\
[\Gamma \vdash \mathsf{Type}_i]^\pi_\gamma &= \mathcal{U}_i \\
[\Gamma \vdash \mathsf{Prop}]^\pi_\gamma &= \{X : X \textit{ is a degenerated } \Lambda\textit{-set}\} \\
[\Gamma \vdash x]^\pi_\gamma &= \gamma(x) \\
[\Gamma \vdash \Pi x:T.U]^\pi_\gamma &= \Pi_{\Gamma\vdash \Pi x:T.U}([\Gamma \vdash T]^\pi_\gamma, [\Gamma(x:T) \vdash U]^\pi_{\gamma,_}) \\
[\Gamma \vdash \lambda_{x:T^\circ.U^\circ} M]^\pi_\gamma &= \downarrow_{\Gamma\vdash \Pi x:T^\circ.U^\circ}([\Gamma(x:T^\infty) \vdash M]^\pi_{\gamma,_}) \\
[\Gamma \vdash \mathsf{app}_{x:T^\circ.U^\circ}(M,N)]^\pi_\gamma &= \uparrow_{\Gamma\vdash \Pi x:T^\circ.U^\circ}([\Gamma \vdash M]^\pi_\gamma)([\Gamma \vdash N]^\pi_\gamma) \\
[\Gamma \vdash \mathsf{nat}^s]^\pi_\gamma &= \mathcal{F}^{(\!|s|\!)_\pi}_{\mathcal{N}}(\bot) \\
[\Gamma \vdash \mathsf{O}]^\pi_\gamma &= (0,\emptyset)
\end{aligned}$$

$$[\Gamma \vdash \mathsf{S}(N)]^\pi_\gamma = (1, [\Gamma \vdash N]^\pi_\gamma)$$

$$[\Gamma \vdash \mathsf{case}_{P^\circ}\ x := M \ \mathsf{of}\ N_0, N_1]^\pi_\gamma = \begin{cases} \bot_{[\Gamma(x\,:\,\mathsf{nat}) \vdash P^\infty]^\pi_{\gamma,\bot}} & \text{if } [\Gamma \vdash M]^\pi_\gamma = \emptyset; \\ [\Gamma \vdash N_0]^\pi_\gamma & \text{if } [\Gamma \vdash M]^\pi_\gamma = (0, \emptyset); \\ \uparrow_{\Gamma \vdash T}([\Gamma \vdash N_1]^\pi_\gamma)(\alpha) & \text{if } [\Gamma \vdash M]^\pi_\gamma = (1, \alpha) \end{cases}$$

$$\text{where } T \equiv \Pi y : \mathsf{nat}.P^\circ\,[x := S(y)]$$

$$[\Gamma \vdash \mathsf{fix}\ f(x : \mathsf{nat}^\star) : U^\star := M]^\pi_\gamma = \epsilon(F, P)$$

where $F \in [\Gamma \vdash \Pi x : \mathsf{nat}^\infty.U^\infty]^\pi_\gamma$, *$P$ is the conjunction of the following properties:*

$$\uparrow(F)\emptyset = \bot_{[\Gamma(x\,:\,\mathsf{nat}) \vdash U^\infty]^\pi_{\gamma,\emptyset}}; \tag{1}$$

$$\uparrow(F)(0, \emptyset) = \uparrow([\Gamma(f : |T|) \vdash M]^\pi_{\gamma,F})(0, \emptyset); \tag{2}$$

$$\uparrow(F)(1, \alpha) = \uparrow([\Gamma(f : |T|) \vdash M]^\pi_{\gamma,F})(1, \alpha), \textit{for all } (1, \alpha) \in \mathcal{N} \tag{3}$$

and we write $|T|$ *for* $\Pi x : \mathsf{nat}.|U|$ *and* $\uparrow$ *for* $\uparrow_{\Gamma \vdash |T|}$.

We write $[\Gamma(x : T) \vdash M]^\pi_{\gamma,\_}$ *as a short hand for*

$$\alpha \in [\Gamma \vdash T]^\pi_\gamma \mapsto [\Gamma(x : |T|) \vdash M]^\pi_{\gamma,\alpha}\ .$$

The conditions imposed on the interpretation of fixpoint construction ensure the stability under μ-reductions. In the main soundness theorem, we prove that the typing rules for fixpoint ensure the existence of a unique function F satisfying the above conditions.

4.3 Interpretation of simple Types.

We define a second (relational) interpretation for simple types. The intention of this second interpretation is to cope with the lack of size annotations in types. Consider the following derivation:

$$\frac{\Gamma(x : \mathsf{nat}^s) \vdash M : \mathsf{nat}^r}{\Gamma \vdash \lambda x : \mathsf{nat}.M : \mathsf{nat}^s \to \mathsf{nat}^r}$$

Note that s and r above could be any size expression. But this size information is not present in the term $\lambda x : \mathsf{nat}.M$. The interpretation of this term is a function in the set $\mathbb{N} \to \mathbb{N}$, specifically $\alpha \in \mathbb{N} \mapsto [M]_\alpha$. To show that the term respects the sizes s and r, we use the relational interpretation of the type. In the case of $\mathsf{nat}^s \to \mathsf{nat}^r$, the relational interpretation, denoted $[\![\cdot]\!]$, is

$$[\![\mathsf{nat}^s \to \mathsf{nat}^r]\!] = \{(f_1, f_2) \in \mathbb{N} \to \mathbb{N} : \alpha < [s] \Rightarrow f_1\alpha = f_2\alpha < [r]\}$$

Then, the interpretation satisfies $([\lambda x : \mathsf{nat}.M], [\lambda x : \mathsf{nat}.M]) \in [\![\mathsf{nat}^s \to \mathsf{nat}^r]\!]$. The relational interpretation can be extended to simple types. Intuitively, the soundness judgment says that if $\Gamma \vdash M : T$, then $([M], [M]) \in [\![T]\!]$.

The relational interpretation also satisfies the contravariance rule. Consider a stage $s' \sqsubseteq s$; the relational interpretation of $\mathsf{nat}^{s'} \to \mathsf{nat}^r$ gives

$$[\![\mathsf{nat}^{s'} \to \mathsf{nat}^r]\!] = \{(f_1, f_2) \in \mathbb{N} \to \mathbb{N} : \alpha < [s'] \Rightarrow f_1\alpha = f_2\alpha < [r]\}$$

Since $[s'] \leq [s]$, we have $[\![\mathsf{nat}^s \to \mathsf{nat}^r]\!] \subseteq [\![\mathsf{nat}^{s'} \to \mathsf{nat}^r]\!]$.

Definition. Let T be a simple type such that $[\Gamma \vdash T^\infty]^\pi_{\gamma_1}$, $[\Gamma \vdash T^\infty]^\pi_{\gamma_2}$, $[\Gamma \vdash T]^\pi_{\gamma_1}$, and $[\Gamma \vdash T]^\pi_{\gamma_2}$ are Λ-sets and that $[\Gamma \vdash T]^\pi_{\gamma_1} = [\Gamma \vdash T]^\pi_{\gamma_2}$. The relational interpretation of T, denoted $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}$, is a Λ-set with a carrier-set included in

$$[\Gamma \vdash T^\infty]^\pi_{\gamma_1} \times [\Gamma \vdash T^\infty]^\pi_{\gamma_2} \ .$$

It is defined as follows: if $\Gamma^\infty \vdash T^\infty : \mathsf{Prop}$, then

$$[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} = [\Gamma \vdash T]^\pi_{\gamma_1} \times [\Gamma \vdash T]^\pi_{\gamma_2};$$

otherwise, it is defined by induction on the structure of T:

- if $T \equiv \Pi x : T_1.T_2$ and $\mathsf{simple}(T_1)$ and $\mathsf{simple}(T_2)$. Assume $[\![\Gamma \vdash T_1]\!]^\pi_{\gamma_1,\gamma_2}$ is a defined Λ-set (denoted by $[\![T_1]\!]$), and for every $(\alpha_1, \alpha_2) \in [\![T_1]\!]$, $[\![\Gamma(x : |T_1|) \vdash T_2]\!]^\pi_{(\gamma_1,\alpha_1),(\gamma_2,\alpha_2)}$ is a defined Λ-set (denoted by $[\![T_2]\!](\alpha_1, \alpha_2)$).
  We define $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} = (X, \models, \bot)$, where

$$\begin{aligned} X = \{(f_1, f_2) \in [\Gamma \vdash T^\infty]^\pi_{\gamma_1} \times [\Gamma \vdash T^\infty]^\pi_{\gamma_2} : \\ (\alpha_1, \alpha_2) \in [\![T_1]\!] \Rightarrow (f_1(\alpha_1), f_2(\alpha_2)) \in [\![T_2]\!](\alpha_1, \alpha_2)\}; \end{aligned} \tag{4}$$

$$\begin{aligned} M \models (f_1, f_2) \iff & N \models_{[\![T_1]\!]} (\alpha_1, \alpha_2) \Rightarrow \\ & \mathsf{app}_{x:T^\circ.U^\circ}(M, N) \models_{[\![T_2]\!](\alpha_1,\alpha_2)} (f_1(\alpha_1), f_2(\alpha_2)) \end{aligned} \tag{5}$$

$$\bot = (\bot_{[\Gamma \vdash T^\infty]^\pi_{\gamma_1}}, \bot_{[\Gamma \vdash T^\infty]^\pi_{\gamma_2}}); \tag{6}$$

- if $T \equiv \mathsf{nat}^s$, we define $[\![\Gamma \vdash \mathsf{nat}^s]\!]^\pi_{\gamma_1,\gamma_2} = ([\Gamma \vdash \mathsf{nat}^s]^\pi_{\gamma_1})^2$;
- otherwise, $\mathrm{SV}(T) = \emptyset$ and we define $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} = ([\Gamma \vdash T]^\pi_{\gamma_1})^2$.

Note that, intuitively, α_1 and α_2 are related in $[\![\mathsf{nat}^s]\!]^\pi$ if they are equal, and the "height" of α_1 is less than $(\!|s|\!)_\pi$.

Given a simple type T, such that $\mathrm{SV}(T) = \emptyset$, there might be more than one clause of the above definition that applies. The following lemma states that the definition does not depend on which clause we use.

Lemma 3. *Let T be a term such that* $\mathsf{simple}(T)$ *and* $\mathrm{SV}(T) = \emptyset$. *If $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}$ is defined, then $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} = ([\Gamma \vdash T]^\pi_{\gamma_1})^2$.*

Figure 3 sums up some properties of the interpretation: stability under weakening, substitution (of stages and terms), and reduction, monotonicity of stage substitution, and soundness of subtyping. We use $\doteq$ to denote Kleene equality: $a \doteq b$ iff a and b are both defined and equal, or if both are undefined.

4.4 Soundness

We extend the relational interpretation of types to contexts in the following way:

$$\begin{aligned} [\![[]]\!]^\pi &= \{(\emptyset, \emptyset)\} \\ [\![\Gamma(x : T)]\!]^\pi &= \{((\gamma_1, \alpha_1), (\gamma_2, \alpha_2)) : (\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi \wedge (\alpha_1, \alpha_2) \in [\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}\} \end{aligned}$$

Below is the main soundness theorem. In the following, we write $[\Gamma \vdash M]^\pi_{\gamma_1,\gamma_2}$ to mean $([\Gamma \vdash M]^\pi_{\gamma_1}, [\Gamma \vdash M]^\pi_{\gamma_2})$.

Term interpretation

Weakening: $[\Gamma\Delta \vdash M]^\pi_{\gamma,\delta} \doteq [\Gamma(z:T)\Delta \vdash M]^\pi_{\gamma,\alpha,\delta}$

Substitution: $[\Gamma \vdash M\,[\imath := s]]^\pi_\gamma \doteq [\Gamma \vdash M]^{\pi(\imath:=(\!|s|\!)_\pi)}_\gamma$

$[\Gamma, \Delta\,[x := N] \vdash M\,[x := N]]^\pi_{\gamma,\delta} \doteq [\Gamma(x:T)\Delta \vdash M]^\pi_{\gamma,[\Gamma \vdash N]^\pi_\gamma,\delta}$

Reduction: $M \to N \Rightarrow [\Gamma \vdash M]^\pi_\gamma \doteq [\Gamma \vdash N]^\pi_\gamma$

Relational interpretation

Weakening: $[\![\Gamma\Delta \vdash U]\!]^\pi_{(\gamma_1,\delta_1),(\gamma_2,\delta_2)} \doteq [\![\Gamma(z:T)\Delta \vdash U]\!]^\pi_{(\gamma_1,\alpha_1,\delta_1),(\gamma_2,\alpha_2,\delta_2)}$

Substitution: $[\![\Gamma \vdash T\,[\imath := s]]\!]^\pi_{\gamma_1,\gamma_2} \doteq [\![\Gamma \vdash T]\!]^{\pi(\imath:=(\!|s|\!)_\pi)}_{\gamma_1,\gamma_2}$

$[\![\Gamma\Delta\,[x := N] \vdash T\,[x := N]]\!]^\pi_{(\gamma_1,\delta_1),(\gamma_2,\delta_2)} \doteq [\![\Gamma(x:U)\Delta \vdash T]\!]^\pi_{(\gamma_1,\nu_1,\delta_1),(\gamma_2,\nu_2,\delta_2)}$ where $\nu_1 \equiv [\Gamma \vdash N]^\pi_{\gamma_1}$, $\nu_2 \equiv [\Gamma \vdash N]^\pi_{\gamma_2}$

Monotony: $s \sqsubseteq r \wedge \imath \;\mathsf{pos}\; T \Rightarrow [\![\Gamma \vdash T]\!]^{\pi(\imath:=s)}_{\gamma_1,\gamma_2} \subseteq [\![\Gamma \vdash T]\!]^{\pi(\imath:=r)}_{\gamma_1,\gamma_2}$

Subtyping: $T \leq U \Rightarrow [\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} \subseteq [\![\Gamma \vdash U]\!]^\pi_{\gamma_1,\gamma_2}$

Fig. 3. Properties of the interpretation

Lemma 4 (Soundness).

1. *If* $\Gamma \vdash M : T$ *and* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$*, then* $[\Gamma \vdash M]^\pi_{\gamma_1,\gamma_2}$, $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}$ *are defined and*
$$[\Gamma \vdash M]^\pi_{\gamma_1,\gamma_2} \in [\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2} \ .$$
2. *If* $\Gamma \vdash T : u$*,* $\mathsf{simple}(T)$*, and* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$*, then* $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}$ *is a defined* Λ*-set.*
3. *If* $\mathsf{WF}(\Gamma)$ *and* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$*, then* $(\gamma_1, \gamma_1) \in [\![\Gamma]\!]^\pi$ *and* $(\gamma_1, \gamma_1) \in [\![\Gamma]\!]^\infty$.

4.5 Strong Normalization

In this section we prove our main result: strong normalization of CIC$\widehat{\ }$.

A substitution is a mapping θ from variables to terms, such that $\theta(x) \neq x$ for a finite number of variables x. We use θ to denote substitutions, and ε to denote the identity substitution. We write $\theta(x \mapsto M)$ for the substitution that gives M when applied to x and $\theta(y)$ when applied to $y \neq x$. We write $M\theta$ for the capture-avoiding substitution of the free variables of M with θ.

Definition 16. *Let* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$*. We define* $\theta \models^\Gamma_\pi (\gamma_1, \gamma_2)$ *by the following clauses:*

$$\frac{}{\varepsilon \models^\Gamma_\pi []} \qquad \frac{\theta \models^\Gamma_\pi (\gamma_1, \gamma_2) \qquad M \models_{[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}} (\alpha_1, \alpha_2)}{\theta(x \mapsto M) \models^{\Gamma(x:T)}_\pi (\gamma_1, \alpha_1), (\gamma_2, \alpha_2)}$$

Lemma 5. *If* $\Gamma \vdash M : T$ *and* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$ *and* $\theta \models^\Gamma_\pi (\gamma_1, \gamma_2)$*, then*
$$M\theta \models_{[\![\Gamma \vdash T]\!]^\pi_{\gamma_1,\gamma_2}} [\Gamma \vdash M]^\pi_{\gamma_1,\gamma_2}$$

Corollary 1. *If $\Gamma \vdash M : T$ then M is strongly normalizing.*

Proof. It is not difficult to see that $\varepsilon \models^{\Gamma}_{\pi} \bot_{\Gamma}$, where $\bot_{\Gamma}$ is the sequence of atomic elements of relational interpretation the types of Γ. The result follows from the previous lemma, and the fact that realizers are strongly normalizing. □

As a consequence of soundness and strong normalization, we can easily derive logical consistency.

Theorem 1. *There is no term M such that $\vdash M : \Pi x : \mathsf{Prop}.x$.*

5 Related Work

The idea of using sized types to ensure termination and productivity was initiated by Hughes, Pareto and Sabry [8]. Abel [1] extended system F^{ω} with sized (co-)inductive types. In turn, CIC$\hat{}_{-}$ is derived from CIC$\hat{}$ [4]. We refer to these papers for further information and examples, and focus on work related to strong normalization.

Our proof of strong normalization closely follows the work of Melliès and Werner [11]. The authors use Λ-sets to develop a generic proof of strong normalization for Pure Type Systems. They avoid the use of inaccessible cardinals, but it is unlikely that the same can be achieved in the presence of inductive types.

Λ-sets were introduced by Altenkirch in [2]. He develops a generic model for the Calculus of Constructions (CC), that can be instantiated with Λ-sets to obtain a proof of strong normalization. He also extends the proof to include one inductive type (trees) at the impredicative level.

There are in the literature several proofs of strong normalization for (non-dependent) typed lambda calculi extended with sized types. We refer the reader to [1,4] for further references.

On dependent types, an extension with type-based termination was first considered by Giménez [6]. However, stages are not explicitly represented, which complicates the definition of mutually recursive functions.

Barras [3] has formalized in Coq a proof of consistency CCω extended with natural numbers. Termination of recursive functions is ensured by a restricted form of sized types, inspired by the work of Giménez. However, it is not possible to express size-preserving functions, which prevents the typing of quicksort.

Blanqui [5] uses sized types to ensure termination of CC extended with higher-order rewrite rules. In our case, we use just $\mathsf{fix}/\mathsf{case}$ instead of rewrite rules. However, he makes some assumptions on the confluence and subject reduction of the combination of rewriting and β-reduction. Nevertheless, it is of interest to see if these techniques can be extended to CIC$\hat{}_{-}$.

Finally, let us mention the work of Wahlstedt [12]. He proves weak normalization of a predicative type theory in the style of Martin-Löf type theory. Termination is ensured using the size-change principle [9]. While this principle is very powerful, his system cannot express size-preserving functions.

6 Conclusions

We presented CIC$\hat{}_{\sqsubset}$, an extension of CIC with a type-based termination mechanism. We have restricted to one inductive type, namely natural numbers, and we have proved that the system is strongly normalizing and logically consistent. The interpretation can be extended to other (positive) inductive types (in the predicative universes). This is a intermediate result towards our goal of proving logical consistency of an extension of CIC with a type-based termination mechanism.

There are some issues related with CIC$\hat{}$ that are present in CIC$\hat{}_{\sqsubset}$ and have not been addressed in this work, namely, global definitions and mutually recursive functions. We have preliminary results in this direction.

Acknowledgments. The authors would like to thank Bruno Barras, Hugo Herbelin, and Benjamin Werner for many discussions on strong normalization and type-based termination.

References

1. Andreas Abel. *A Polymorphic Lambda-Calculus with Sized Higher-Order Types*. PhD thesis, Ludwig-Maximilians-Universität München, 2006.
2. Thorsten Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, University of Edinburgh, November 1993.
3. Bruno Barras. Sets in coq, coq in sets. 1st Coq Workshop, August 2009.
4. Gilles Barthe, Benjamin Grégoire, and Fernando Pastawski. CIC$\hat{}$: Type-based termination of recursive definitions in the Calculus of Inductive Constructions. In Miki Hermann and Andrei Voronkov, editors, *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2006.
5. Frédéric Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In Vincent van Oostrom, editor, *RTA*, volume 3091 of *Lecture Notes in Computer Science*, pages 24–39. Springer, 2004.
6. Eduardo Giménez. Structural recursive definitions in type theory. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 397–408. Springer, 1998.
7. Benjamin Grégoire and Jorge Luis Sacchini. On strong normalization of the Calculus of Constructions with type-based termination. To appear in LPAR, 2010.
8. John Hughes, Lars Pareto, and Amr Sabry. Proving the correctness of reactive systems using sized types. In *POPL*, pages 410–423, 1996.
9. Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The size-change principle for program termination. In *POPL*, pages 81–92, 2001.
10. Zhaohui Luo. *Computation and reasoning: a type theory for computer science*. Oxford University Press, Inc., New York, NY, USA, 1994.
11. Paul-André Melliès and Benjamin Werner. A generic normalisation proof for pure type systems. In Eduardo Giménez and Christine Paulin-Mohring, editors, *TYPES*, volume 1512 of *LNCS*, pages 254–276. Springer, 1996.
12. David Wahlstedt. *Dependent Type Theory with Parameterized First-Order Data Types and Well-Founded Recursion*. PhD thesis, Chalmers University of Technology, 2007. ISBN 978-91-7291-979-2.